View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Universidade do Minho: RepositoriUM

Integration of Flight Simulator 2002 with an epidemic multicast protocol

M.João Monteiro
Universidade de Lisboa
mjmonteiro@di.fc.ul.pt

José Pereira
Universidade do Minho
jop@di.uminho.pt

Luís Rodrigues
Universidade de Lisboa
ler@di.fc.ul.pt

***Abstract*— Multi-player games are increasingly popular in the Internet. This growing interest results in the need to support a high number of participants, which raises the issue of scalability, namely in what regards the ability to offer good performance in such a large-scale setting. This paper addresses the issue of designing middleware solutions to support large-scale multi-player applications. In particular, we are interested in studying the feasibility of using epidemic multicast protocols for information dissemination in multi-player games based on a peer-to-peer architecture. We have integrated a comercial multi-player game, the Microsoft Flight Simulator 2002, with an epidemic multicast protocol appropriate to this kind of applications, the NEEM. The paper describes how this integration was achieved and presents evaluation results of the resulting prototype.**

Keywords: Epidemic multicast, peer-to-peer architecture, multi-player games.

I. Introduction

Multi-player applications are more and more common in the Internet today. Such applications have generated interest from users and from hosting companies who see them as new sources of income and useful advertisement vehicles. Multi-players games are one of the most popular instances of these applications. Due to the large number of users that games can attract, these applications raise interesting scalability problems. One should note that the users of these games are not confined to players, since on-line games are becoming spectators' games [2], with spectators actively participating in the game through chatting or even through betting[1].

A common requirement for multi-players games is the necessity of multicasting updates among users, on a regular basis. Unfortunately, it is extremely hard, if not impossible, to enforce reliable multicast in a scalable way. Some protocols, such as RMTP [10], generate a large number of acknowledgments that must be received and processed, a task that may quickly overcharge the sender. Even with more sophisticated acknowledgment mechanisms [7], messages have to be buffered and retransmitted until all recipients acknowledge their reception or are declared failed. So, when a receiver is slower, messages accumulate in the sender buffer, which will inevitably be forced to adjust its throughput, thus causing the whole group to be slower.

Probabilistic multicast protocols, also called, gossip-based or epidemic, overcome the scalability limitations mentioned above. They support the efficient dissemination of data among a large number of nodes while providing a probabilistic guarantee of delivery [8], [1], [9], [3]. These protocols present an interesting compromise between the reliability and the scalability requirements of multi-player games. By using this type of protocols, it is possible to assure throughput stability even for very large groups with heterogeneous behaviour, as the load required to ensure reliability is evenly spread across all members and no single perturbed node can block senders. Among these protocols, we highlight NEEM [14]. This protocol combines the use of TCP at the transport layer (for a network-friendly operation) with an efficient buffer management that allows purging of messages that are already obsolete. NEEM is particularly tailored to support large-scale multi-player games.

The present paper describes the effort of integrating a commercial multi-player game, the Microsoft Flight Simulator 2002, with the NEEM epidemic multicast protocol. The goal of this work is to validate the usefulness of this kind of protocols to solve the problems of scalability that are raised when a very large number of participants is involved in the game.

The rest of the paper is structured as follows: For self-containment, Section II introduces the Flight-Simulator

This work has been partially supported by FCT project RUMOR (POSI/40088/CHS/2001) and Microsoft Research Grant (2001-39).

Contact: Faculdade de Ciências da Universidade de Lisboa, Dep. de Informática. Bloco C5, Campo Grande 1749-016 Lisboa, Portugal.

[1]As an example visit `http://youplaygames.com`.

game and Section III makes an overview of the NEEM protocol. Section IV describes the integration of these components and Section V includes the evaluation of the resulting prototype. Finally, Section VI concludes the paper.

II. Microsoft Flight Simulator 2002

The game we have selected to perform and validate the integration of multi-player games with probabilistic multicast protocols is the Microsoft Flight Simulator 2002 (FS2002). In the following sections we offer a general overview of the FS2002 operation, and we discuss some peculiarities that are relevant to the integration work. The scalability problems of multi-player games are also discussed in the concrete context of FS2002.

A. Game Overview

FS2002 is a realistic flight simulator game which supports multi-player operation both on local area networks and on the Internet. Each player controls a single aircraft in a virtual world, shared by all the players. Several kinds of aircrafts and flight charts are made available to each player. In a multi-player session there is a host that selects the scenario in which the session will take place and which is responsible for maintaining the group membership. If the current host fails, a new node takes over this role such that the game can proceed. The users can interact with each other through text messages, creating a typical ”chat” environment.

The FS2002 implementation on the Windows operating system uses the DirectPlay API [11]. This is a message passing protocol often used to implement multi-player games in Windows. DirectPlay, also offers a membership protocol. FS2002 relies on a peer-to-peer architecture in which the state of the game is replicated on every participant. For this purpose, each player maintains the authoritative state of the locally controlled aircraft and periodically updates other participants by sending information about its position and velocity [12]. The packet that carries the state updating is presented in Fig 1. The velocity vector is represented by its three components (v_lat, v_lon, v_alt). The fields corresponding to the position of the aircraft are the last six fields. To represent the position of the aircraft, latitude, longitude and altitude values are disseminated linearized into high and low order bit fields, which are identified by suffixes $\_M$ and $\_m$, respectively, in Fig 1.

B. Performance restrictions

In nowadays Internet, FS2002 participants access the network through residential connections such as ADSL, cable or even traditional analog modems. Even broadband connections are asymmetric and have restricted uplink capacities. This makes the use of a peer-to-peer architecture to disseminate all updates extremely challenging as a single node does not have enough bandwidth to disseminate the information to all other nodes. The use of probabilistic multicast protocols alleviates this problem since each node only has to communicate with a few other nodes to achieve multicast. However, even with these protocols, the restricted bandwidth available can generate congestion, inducing message losses. It is worth noting that a centralized solution also raises scalability problems on the server side.

Let us analyze the communication requirements of a game such as FS2002 in a concrete scenario. Each player multicasts 4 state update messages each second, carrying 60 bytes of payload each. When players are connected using V.90 modems (56 kbps downlink, 33.6 kbps uplink) a maximum of 17 peers can be contacted for each state update. As each player is responsible for multicasting its updates to all spectators, this means that 17 is in fact the total number of players plus spectators. Even with broadband connections, the uplink is lower than the available downlink bandwidth, thus limiting the number of destinations. This can be mitigated by relying on a centralized server with a more expensive high bandwidth network connection. However, this does not entirely solve the problem, as a hosting service is also expected to host more than one simultaneous instance of the game. The amount of traffic imposed on the server is still proportional to the total number of spectators and thus the cost of such service grows linearly with the bandwidth used.

In this paper, we propose a different architecture which consists in using an epidemic multicast protocol that allows the spectators to cooperate in a decentralized dissemination of the game updates. With this approach we avoid the use of a centralized server and we take the best advantage of the peer-to-peer architecture, supported by DirectPlay.

III. The NEEM protocol

This work uses a new probabilistic multicast protocol, called NEEM [14], that was conceived to operate in

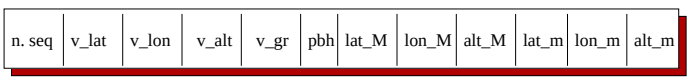

Fig. 1. Update packet.

networks with the same features as those where multi-player games, like FS2002, operate. The next paragraphs describe the NEEM architecture and its main components: the epidemic dissemination mechanisms, the buffer management and the use of TCP.

NEEM was implemented using Appia [13]. Appia is a framework that supports the composition and execution of micro-protocols. Each Appia module is a micro-protocol and is responsible for ensuring certain properties. These modules are independent and can be combined, building a stack configured with the desired properties. Fig 2 presents the stack that implements NEEM. Each layer of the Appia stack is an essential NEEM component, that will be described in the following paragraphs.

a) Epidemic Multicast: The top layer of the stack is the NEEM's epidemic multicast mechanism [4], that works as follows: A message is initially tagged with the maximum number of rounds r and then forwarded to distinct f other nodes chosen randomly. On reception, the number of rounds is decremented. When this counter reaches zero, the message is discarded. In the other cases, the message is forwarded again to another f nodes. Delivery happens when a new message is received by a node. The guarantees offered by the protocol depend on appropriate configuration of r and f [8], [9].

The membership protocol is itself based on gossip and keeps at each node a list of other locally known nodes [3], [5]. This list is a partial view of the entire group and, as it has been shown, this is sufficient for gossiping to succeed, even if this list is much smaller than the entire group. Upon each gossip round, the identification of some locally known nodes is piggybacked on data messages. When a message is received, the local list is updated with the nodes in the arriving list. If the size of the local list overcomes a certain pre-defined limit, some nodes are removed randomly.

It is worth noting that for a group of n participants, the size of the partial local view, as well as the parameters

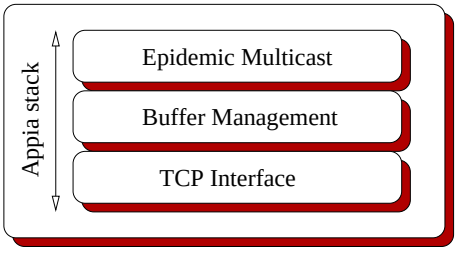

Fig. 2. NEEM.

f and r are of the order of magnitude of $\log n$, which allows the existence of very large groups.

b) Buffer management: The layer responsible for the buffer management stores the received messages that will be gossiped in subsequent rounds. For short bursts of incoming messages, buffering alone is enough to spread the load in time and thus to avoid message losses. For continued loads, the buffers are eventually exhausted. To prevent a single perturbed node to degrade overall performance, the epidemic multicast protocol cannot wait until buffer space is available. The only option is to select a message to be discarded. This selection is done by combining the following strategies:

- *Semantic purging:* A message that has been recognized as obsolete is discarded. In contrast with other purging policies, this strategy can be applied even if the buffer is not full. This has the potencial advantage of reducing average buffer occupancy and thus of lower latency.
- *Age-based purging:* The message that has been relayed more times is discarded. This strategy can be used even if no obsolete messages are discovered and can still offer some performance advantages.
- *Random purging:* A message is selected at random to make up room for each newly arrived message. This strategy is used as a last resort.

An important aspect in the design of the NEEM's interface is how to allow the application convey the required semantic knowledge to the protocol (to allow the identification of obsolete messages) while at the same time retaining the generality of the implementation. The proposed implementation is to associate a small bitmap to each message. If the i^{th} bit is set in the bitmap of message n, then the message with sequence number $n-i$ is considered obsolete.

c) Interface with TCP: The bottom layer is the interface of Appia's stack with the TCP implementation of the underlying operating system. The choice of using TCP relies on the existence of the congestion control mechanism which allows a better usage of the available bandwidth. The available bandwidth in each node is divided by several TCP connections (i.e. as many as the size of the local membership list), so it is possible to reduce the size of the buffers and, therefore, is possible to reduce the latency and the total of resources used [6].

IV. Integration

The integration of the FS2002 with NEEM has exploited the fact that the game is based on the DirectPlay API. We have implemented an adaptation layer, called

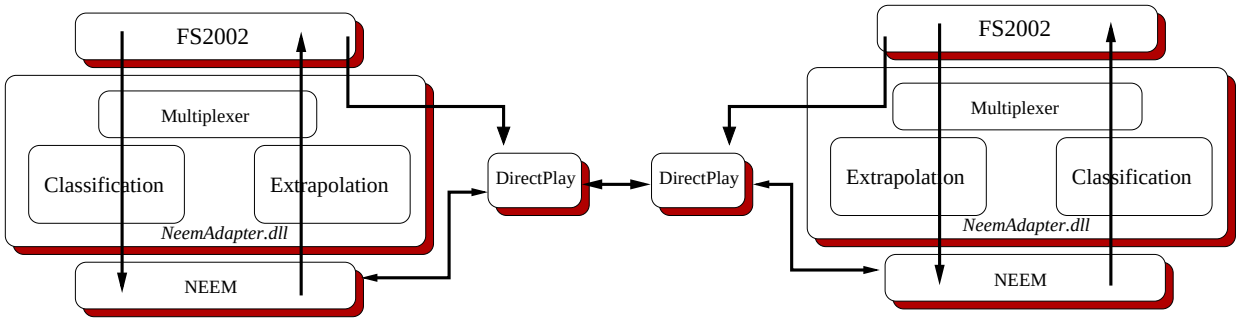

Fig. 3. Integration architecture.

NeemAdapter, a dynamic library that intercepts the calls to the original library *dplayx.dll*. The NeemAdapter intercepts all data messages and forwards the remaining control operations to the original library without further processing. The state update messages have an additional processing both on sending and reception. Outbound traffic is classified and inbound traffic is pre-processed by an extrapolation component. The NeemAdapter is also responsible for transforming the peer-to-peer update multicast of FS2002 in a call to the NEEM protocol. These components are described in the following sections. The Fig 3 presents the integration's architecture.

The NEEM protocol was complemented with an interface layer, at the top of the Appia stack, which is responsible for the interface between the Java[2] implementation of NEEM and the C++ implementation of NeemAdapter. We have also replaced the bottom layer of the NEEM, the TCP layer, by a layer that interfaces with DirectPlay, which in turn calls the native TCP/IP implementation of the operating system. These changes to the Appia stack of the NEEM protocol are depicted in Fig 4.

[2]The Java language was used for rapid prototyping. In a comercial solution the performance restrictions of Java would have to be considered.

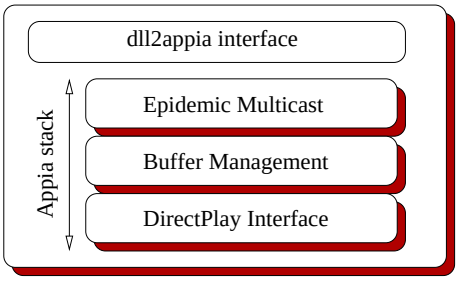

Fig. 4. NEEM adjusted.

A. Multiplexer

FS2002 relies on a peer-to-peer architecture, where each player is responsible for disseminating the update of its position and velocity to all the other participants (players and spectators). Without our adaptation layer, this would be done point-to-point to each of the destinations. In order to benefit from the presence of the epidemic multicast protocol, the NeemAdapter transforms each set of point-to-point calls into a single call to the NEEM protocol. This, in turn, implements an epidemic multicast that, as noted before, can be performed without requiring the sender to communicate directly with all other nodes.

B. Classification

One of the essential issues when using NEEM is how to encode the message semantics. Each of the updates to be sent by a player are submitted to a classification process, where they are related to the previous n updates[3]. In order to do so, the NeemAdapter keeps a list with the last n updates sent. By comparing an update with the ones before, the classification component identifies the previous messages that become obsolete and builds the bitmap associated to the new update. As a result, the buffer management layer only needs to process these bitmaps and remains completely unaware of the content of FS2002's messages.

The rule used to decide when a message makes other obsolete is based on the observation of the distance between both updates: if both messages are referring to positions that differ less than a pre-defined factor, then the elder one is considered obsolete; otherwise, no message is classified as obsolete. So, a message m_j makes m_i obsolete if:

- $j > i$ and

[3]The parameter n depends on the dimension chosen for the bitmap.

- $\forall k : i < k \leq j$, the velocity vector of m_k differs from that of m_i by less than a factor F, more precisely: $|\vec{v_{m_k}} - \vec{v_{m_i}}| \leq F * |\vec{v_{m_i}}|$

When applied to FS2002 traffic, this rule results in, with $F = 0.01$, having a share of 46% of all messages become obsolete shortly after being sent. With $F = 0.02$ this share rises to 72%. These numbers disregard messages other than updates, whose number is relatively insignificant.

C. Extrapolation

The NeemAdapter intercepts all messages received by DirectPlay, with FS2002 as destination. The messages with the state updates arrive with a constant interval between them of approximately 320ms. The NeemAdapter keeps a timeout for each of the aircrafts in the session (except itself), in order to compensate for lost or delayed messages in the probabilistic protocol, by extrapolating these messages from the past positions.

The NeemAdapter keeps the last update received from each aircraft and, if the timeout expires and no new update is received, the extrapolation process is initiated based on the information kept. This process extrapolates a new position from the position of the last update stored and assumes that the velocity remains the same. It then builds a new update packet with this extrapolated information, that is sent up to FS2002. This extrapolated packet is kept in the NeemAdapter, replacing the last one, like any original update coming from the network.

V. Evaluation

Since we do not have the resources to build a controlled experiment with hundreds of FS2002 spectators, we have considered the real traces of the FS2002 and extracted an abstract model of the traffic pattern of FS2002 that was used to feed simulation. Traffic is collected, in the prototype, by intercepting requests at the DirectPlay API. The use of simulation has also the advantage of allowing comparisons between protocols, when submitted exactly to the same traffic.

A. Simulation model

The model used simulates the NEEM behaviour in order to identify which messages are lost when the system is configured with the traffic extracted from FS2002. This model has a set of processes that execute the NEEM protocol, forwarding, storing and delivering messages. The membership mechanism is the only one that is not simulated. Each node is connected to a subset of distinct nodes chosen randomly which constitute its local membership. This membership is generated once before running all simulations. The network is simulated by a set of queues that connect all the participating nodes. Each node then divides the available bandwidth by the connections to the processes it knows.

The results presented in this paper were achieved using 500 nodes. Each point-to-point connection has an associated buffer with capacity for storing 10 messages, in the buffer management layer. For the epidemic multicast, the local membership of each node has 12 entries, the messages are sent to 6 destinations and each messages is retransmitted 6 times. The network is configured such that each link has a 56kbps bandwidth both for downlink and uplink and a latency of 25ms, in the worst case. A justification for these values can be found in [8]. Each simulation run for 300s. The first 100s and the last 50s are discarded to avoid transient states.

The traffic pattern of FS2002 can be generalized as follows. Traffic is composed of an amount of traffic that never becomes obsolete. The remaining share can be divided in as many chains as the aircrafts in the session. We say that each message in a chain is made obsolete by its successor with a probability of 46% – having $F = 0.01$.

B. Results

The NEEM protocol [**?**] is a recent proposal of an epidemic protocol that supports end-to-end congestion control, by using connection-oriented transport connections, and ensures throughput stability by using an innovative buffer management mechanism that discards messages on overflow. The Fig 5 compares the performance of

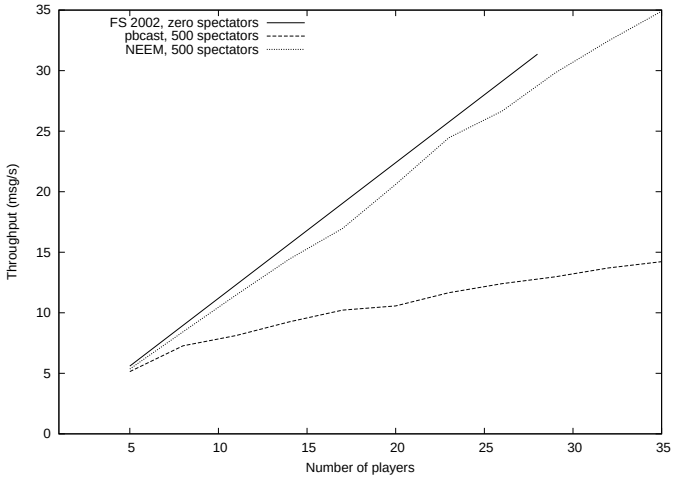

Fig. 5. FS2002 performance.

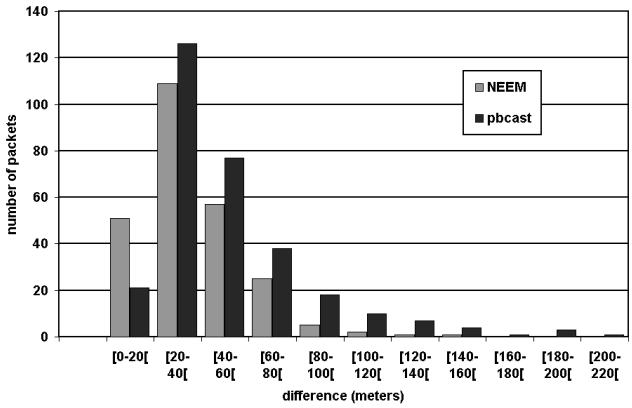

Fig. 6. Routes comparison.

the original FS2002 without spectators with FS2002 plus NEEM with 500 spectators. Considering $F = 0.02$, the figure presents the throughput of the messages that never become obsolete. In the considered scenario, the original decentralized protocol of FS2002 makes it possible to deliver every message successfully until 28 players. It's worth noting that the multicast is made by each of the players, which implies that 28 is the total number of participants (players and spectators). So, the maximum number of spectators that the FS2002 can support is 27, in the extreme case where there is only one player.

With NEEM, the update multicast is distributed. So, it's possible to support a high number of spectators independent of the number of players. The Fig 5 shows how it is possible to deliver almost all messages that never become obsolete, in the presence of 500 spectators. In contrast, a traditional epidemic multicast protocol – pbcast [8] – will loose messages that never become obsolete.

The impact of the message losses is visible when we compare the extrapolated route from both protocols with the original route. The Fig 6 presents such comparison. Using NEEM protocol the difference between the routes is never larger than 100m and most packets that suffer extrapolation present a difference around 20m, resulting in 32m of average. In contrast, the route extrapolated from pbcast implies greater errors, resulting in an average difference of 49m.

VI. Conclusions

This paper presents the integration of a multi-player game, the Microsoft Flight Simulator 2002, with the protocol NEEM, which combines epidemic multicast with message semantics. The evaluation of this integration is done through implementation and simulation. The results achieved show that, in contrast with the original protocol used by the game that only supports few spectators, is possible to disseminate the game state by hundreds of spectators. The use of message semantics allows the support of hundreds of spectators without having to reduce the number of concurrent players, as necessary when using a tradicional epidemic multicast protocol.

References

[1] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2), May 1999.

[2] S. Drucker, L. He, M. Cohen, C. Wong, and A. Gupta. Spectator games: A new entertainment modality of networked multiplayer games. Technical report, Microsoft Research, 2002.

[3] P. Eugster, R. Guerraoui, S. Handrukande, A.-M. Kermarrec, and P. Kouznetsov. Lighweight probabilistic broadcast. In *IEEE Intl. Conf. on Dependable Systems and Networks (DSN)*, 2001.

[4] S. Formigo, J. Pereira, and L. Rodrigues. Difusão probabilista com fiabilidade semântica. In *Actas da 5^a Conferência sobre Redes de Computadores*, Faro, Portugal, September 2002.

[5] A.J. Ganesh, A.-M. Kermarrec, and L. Massoulie. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, February 2003.

[6] A. Goel, C. Krasic, K. Li, and J. Walpole. Supporting low latency TCP-based media streams. In *Intl. Ws. on Quality of Service (IWQoS'2002)*, 2002.

[7] K. Guo. *Scalable Message Stability Detection Protocols*. PhD thesis, May 1998.

[8] M. Hayden and K. Birman. Probabilistic broadcast. Technical Report TR96-1606, Cornell University, Computer Science, 1996.

[9] A.-M. Kermarrec, L. Masssoulié, and A. Ganesh. Reliable probabilistic communication in large-scale information dissemination systems. Technical Report 2000-105, Microsoft Research, 2000.

[10] J. Lin and S. Paul. RMTP: A reliable multicast transport protocol. In *IEEE Conf. Computer Communications (INFOCOM)*, 1996.

[11] Microsoft Corp. *DirectPlay 8.1*, 2002.

[12] Microsoft Corp. *Microsoft Flight Simulator 2002 Software Development Kit – Multiplayer/Flight instructor*, 2002.

[13] H. Miranda, A. Pinto, and L. Rodrigues. Appia, a flexible protocol kernel supporting multiple coordinated channels. In *IEEE Intl. Conf. Distributed Computing Systems (ICDCS)*, 2001.

[14] J. Pereira, L. Rodrigues, M. J. Monteiro, R. Oliveira, and A.-M. Kermarrec. Neem: Network-friendly epidemic multicast. In *Proceedings of the 22th IEEE Symposium on Reliable Distributed Systems (SRDS'02)*, page (accepted for publication), Florence,Italy, October 2003.